Programming Reconfigurable Heterogeneous Computing Clusters Using MPI With Transpilation

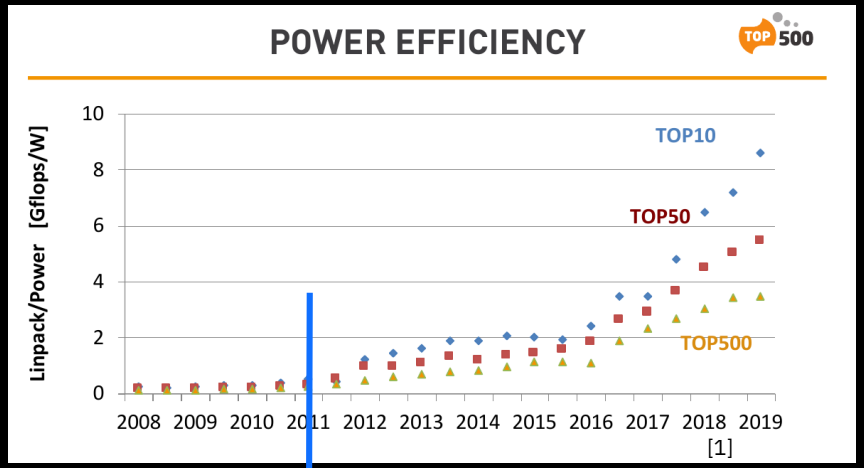
Burkhard Ringlein[1,2], Francois Abel[1], Alexander Ditter[2], Beat Weiss[1], Christoph Hagleitner[1], and Dietmar Fey[2]

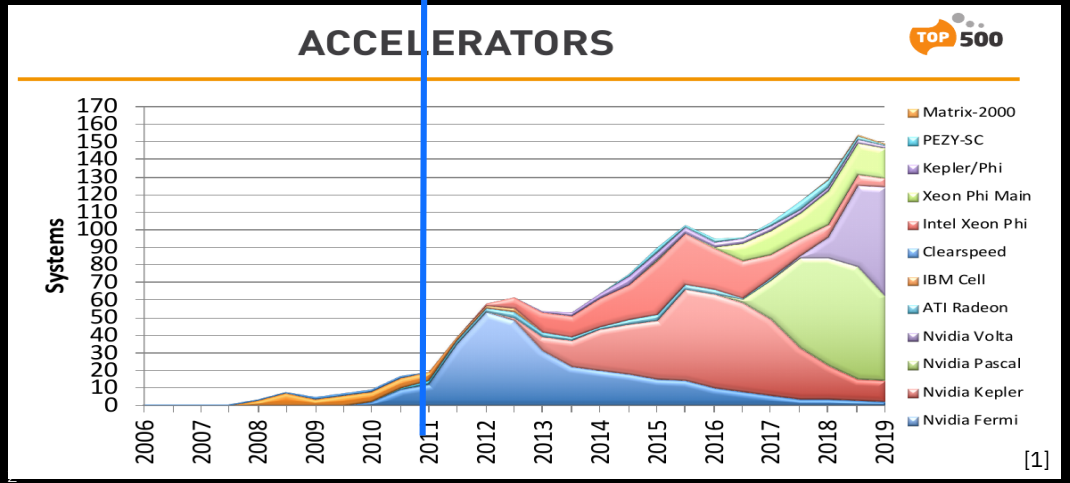
[1]IBM Research Europe
Zurich, Switzerland

[2]Department of Computer Science
Computer Architecture
Friedrich-Alexander University Erlangen-Nürnberg

H2RC20 – 2020-11-13

“Power efficiency → compute power”





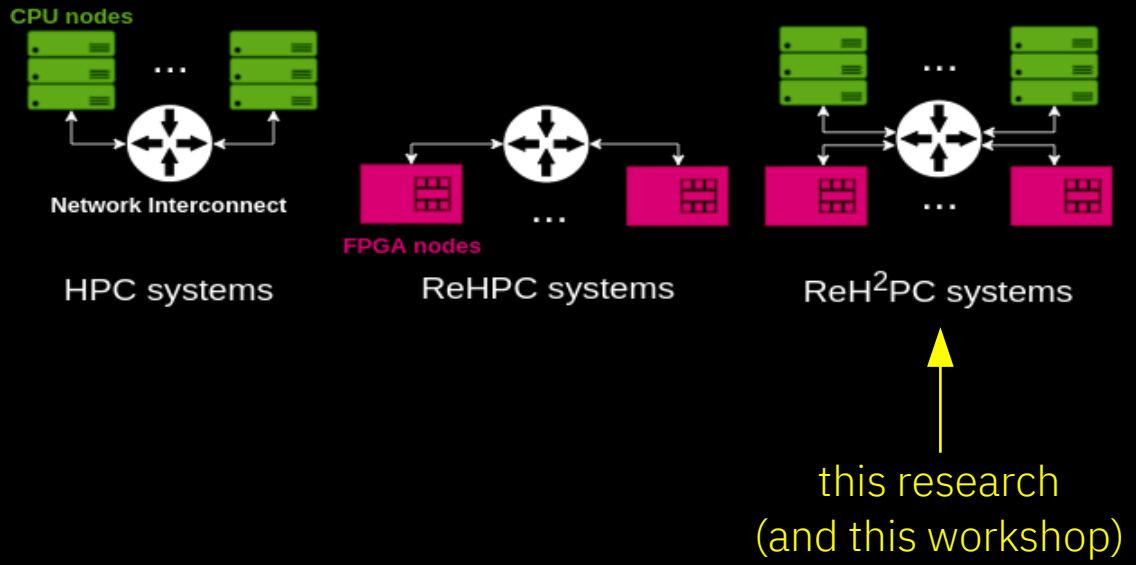
- end of Dennard scaling
- slowdown of Moore’s law
- → HPC needs more and more accelerators

→ actually, *Heterogeneous* HPC (H^2PC)

Reconfigurable Heterogeneous HPC (ReH²PC)

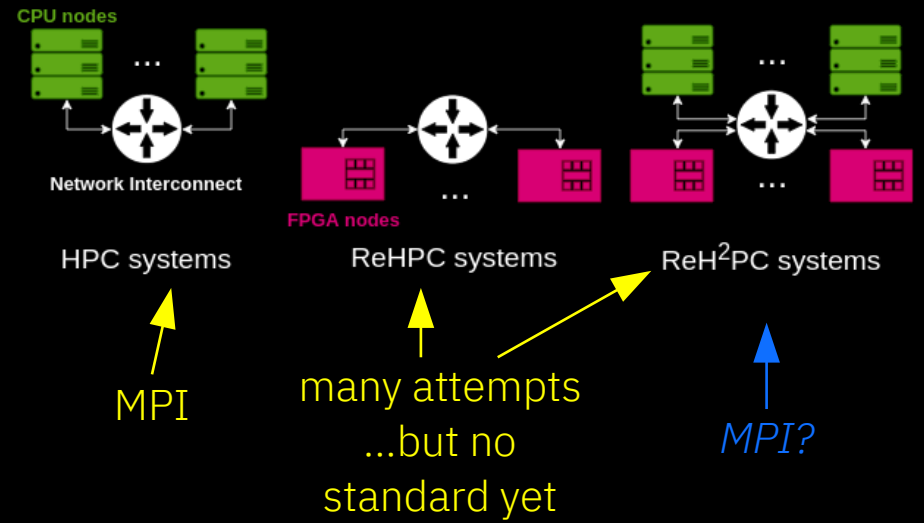
Today's HPC systems can be classified into three classes:

1) HPC: only CPU servers
2) ReHPC: only FPGAs
3) ReH²PC: CPUs + FPGAs



Programming Standards for Heterogeneous HPC

- MPI is the de-facto standard for HPC
- the presence of a standard interface accelerates both:
  platform and app development
- there is no similar established standard for ReHPC or ReH²PC
- since FPGAs barely come without CPUs, we focus on ReH²PC

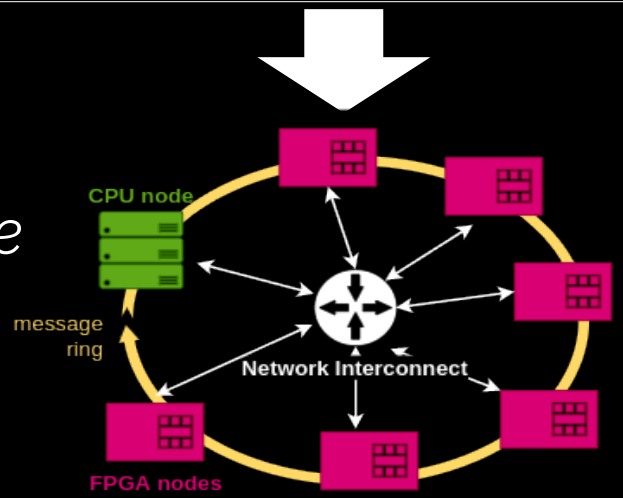


→ *Goal: bring MPI to ReH²PC*

→ Goal: bring MPI to ReH²PC = *one source of code, no modifications*

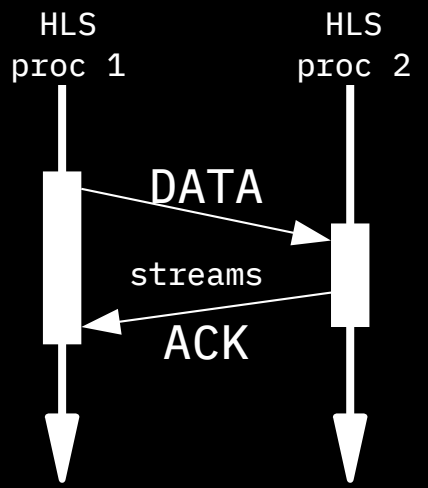
```
int msg[1];
int next_node = (rank + 1) % size;
int previous_node = rank -1;
if(rank == 0) {
    msg[0] = 0xcaffee;
    MPI_Send(&msg[0], 1, MPI_INTEGER, 1, 0, MPI_COMM_WORLD);
    MPI_Recv(&msg[0], 1, MPI_INTEGER, size-1, 0, MPI_COMM_WORLD, &status);
} else {
    MPI_Recv(&msg[0], 1, MPI_INTEGER, previous_node, 0, MPI_COMM_WORLD, &status);
    MPI_Send(&msg[0], 1, MPI_INTEGER, next_node, 0, MPI_COMM_WORLD);
}
```

Heterogeneous MPI message ring example



→ It's about offloading both computing and communication to FPGAs

Why MPI for ReH²PC

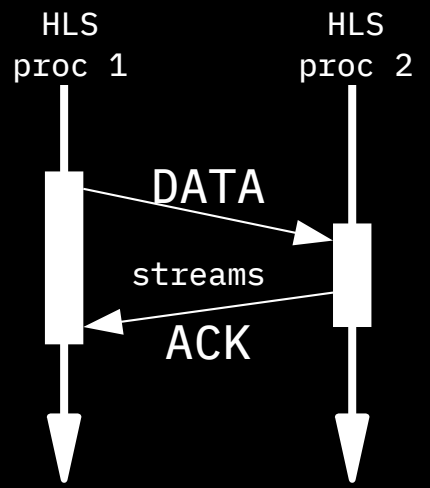


- we can learn from proven concepts
- widely established, thousands of applications
- using HLS design flow:

MPI C → HLS → RTL → partial bitstream

- communication model is similar to how HDL or HLS processes communicate

Why MPI for ReH²PC (2)



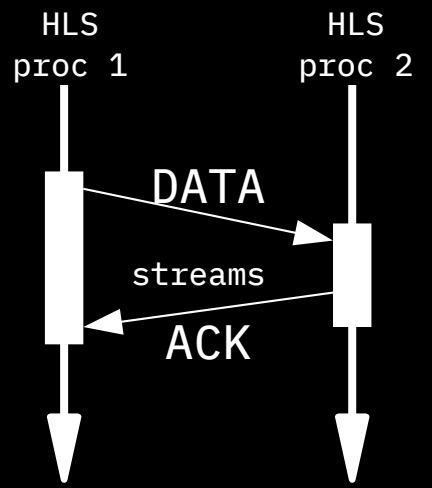
- MPI is based on the BSP model:
  - each node is single threaded (pure MPI, no OpenMP etc.)
  - computation and communication is clearly separated

```
void MPI_Send(void* data, int count, MPI_Datatype datatype,
    int destination, int tag, MPI_Comm communicator);

void MPI_Recv(void* data, int count, MPI_Datatype datatype,
    int source, int tag, MPI_Comm communicator, MPI_Status* status);
```

- the explicit MPI semantic can be easily turned to stream-like read and writes

Difficulties using MPI for ReH²PC



1) “compute first, send later”
   - does not fit parallel processing of FPGAs
   - → “compute while communicating”
2) Single Program, Multiple Data
   - wastes FPGA logic
   - → Multiple Program, Multiple Data

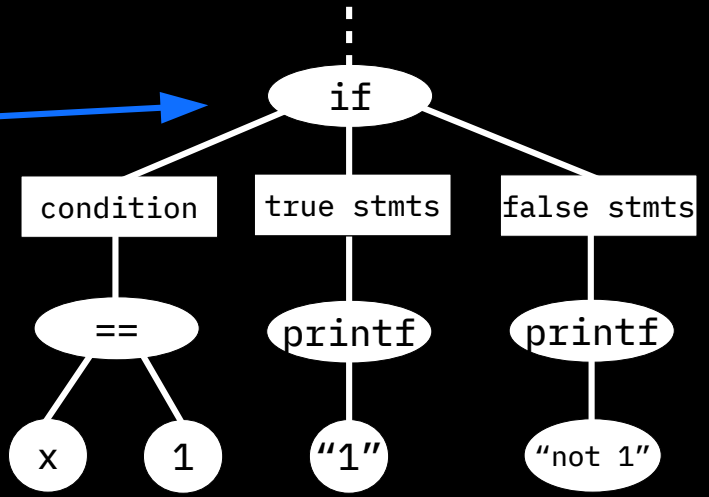
→ MPI for ReH²PC requires:

1) we need a transpiler: C/MPI → HLS
2) we need the *cluster run-time description* at compile-time

Transpilation process

- `cluster description` and `MPI C` as input
- analyze and replace collective routines
- split SPMD into MPMD
  - insert rank and size
  - constant folding
  - evaluate partial AST (Abstract Syntax Tree) for each rank
- optimize collective routines
- generate HLS code

```
{
  "nodes": {
    "cpu"  : [0, 1],
    "fpga" : "2 – 33"
  }
}
```



Transpilation of SPMD to MPMD

clearly defined by MPI syntax & semantic
→ “obvious” to transpile

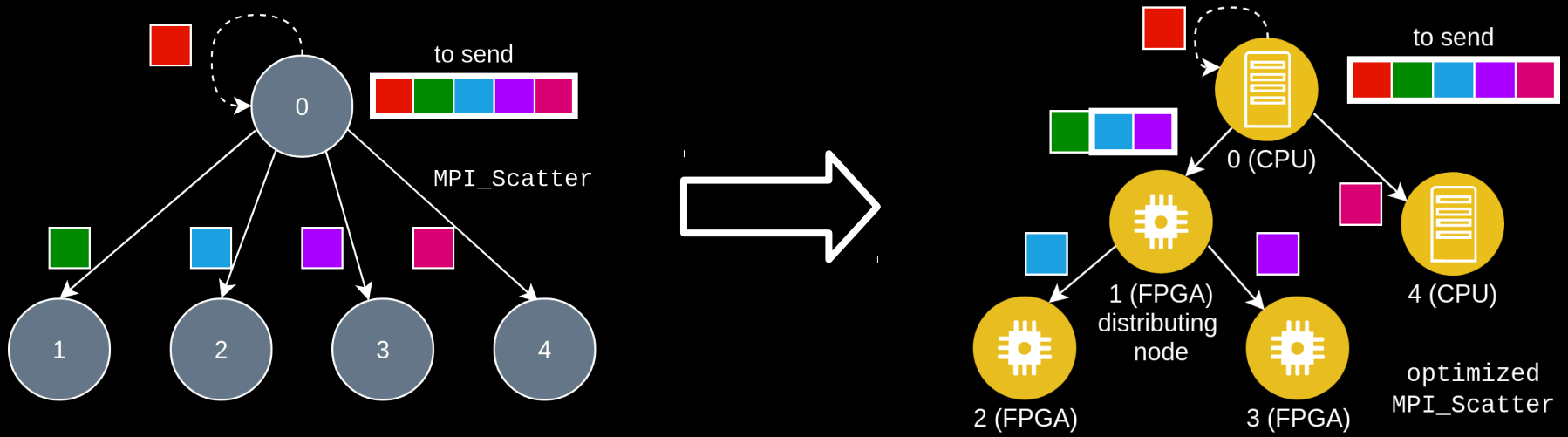
```
....
int rank, size;

MPI_Comm_rank(MPI_COMM_WORLD, &rank );
MPI_Comm_size(MPI_COMM_WORLD, &size );
....
int msg[1];
int next_node = (rank + 1) % size;
int previous_node = rank -1;
if(rank == 0){
    msg[0] = 0xcaffee;
    MPI_Send(&msg[0], 1, MPI_INTEGER,1, 0, MPI_COMM_WORLD);
    MPI_Recv(&msg[0], 1, MPI_INTEGER,size-1, 0, MPI_COMM_WORLD, &status);
} else {
    MPI_Recv(&msg[0], 1, MPI_INTEGER,previous_node, 0, MPI_COMM_WORLD, &status);
    MPI_Send(&msg[0], 1, MPI_INTEGER, next_node, 0, MPI_COMM_WORLD);
}
....
```

```
{
  "nodes": {
    "cpu"  : [0],
    "fpga" : "1 - 6"
  }
}
```

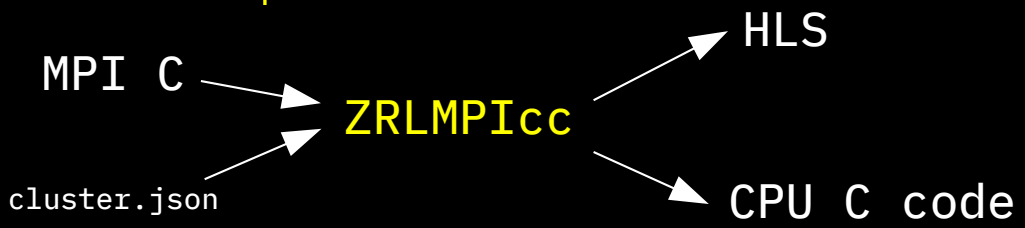
Tree Optimizations

- in general: speed up computation by parallelization of communication
- ReH^{2}PC: leverage lower network latency of FPGAs



The ZRLMPI Framework

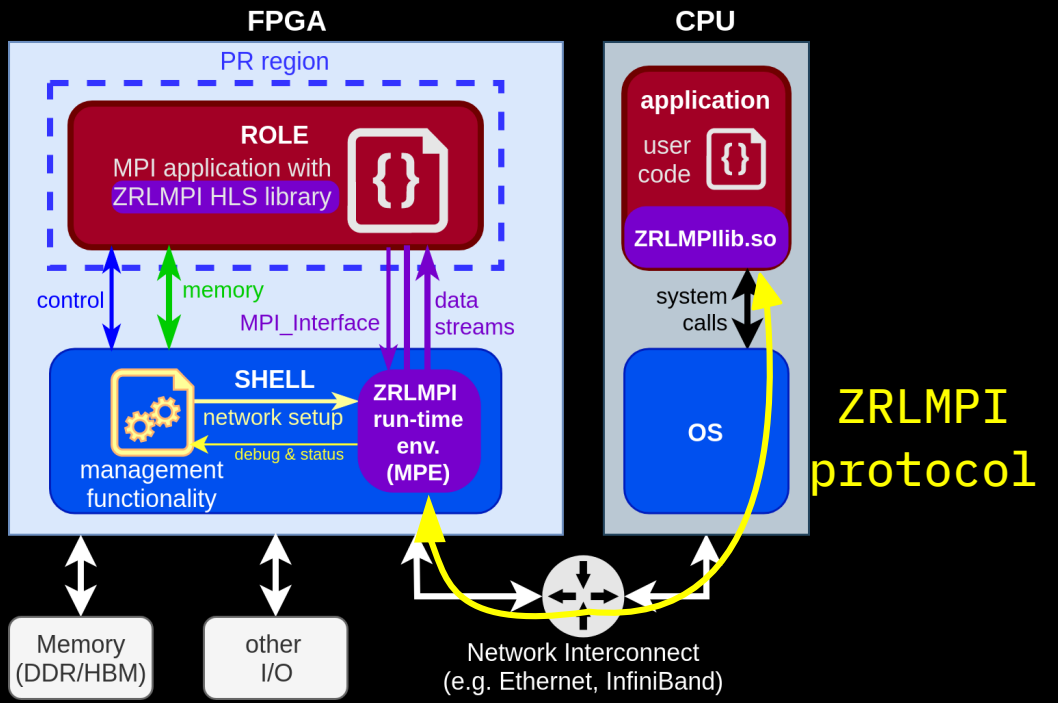
- Transpiler



- `HLS library + MPE`:
  The FPGA run-time environment

- `ZRLMPIlib.so`:
  The CPU run-time environment

```
{
  "nodes": {
    "cpu"  : [0, 1],
    "fpga" : "2 - 33"
  }
}
```

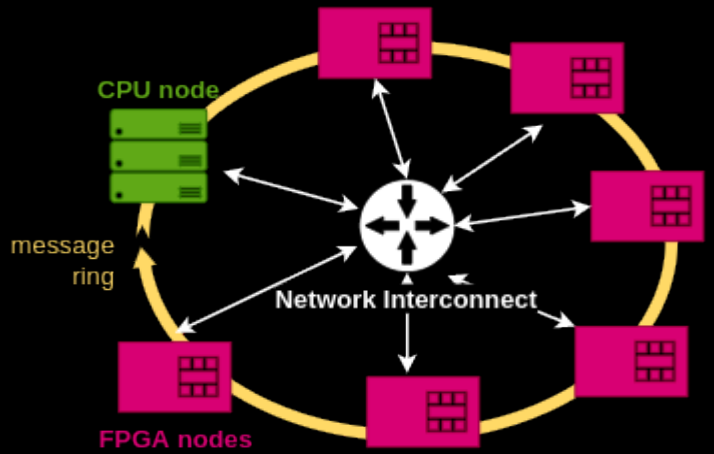


ZRLMPIrun: One-click deployment

host IP
partial bitstream id
software binary
# of FPGAs
software rank

```
$ ZRLMPIrun new udp 10.0.47.11 0ddb12b2-8459-4843-b339-236b2b92b59f 6 ./msg_ring_SW 0
using udp
setting up cluster…
verify network…
start MPI…
....
```

Heterogeneous MPI message ring example



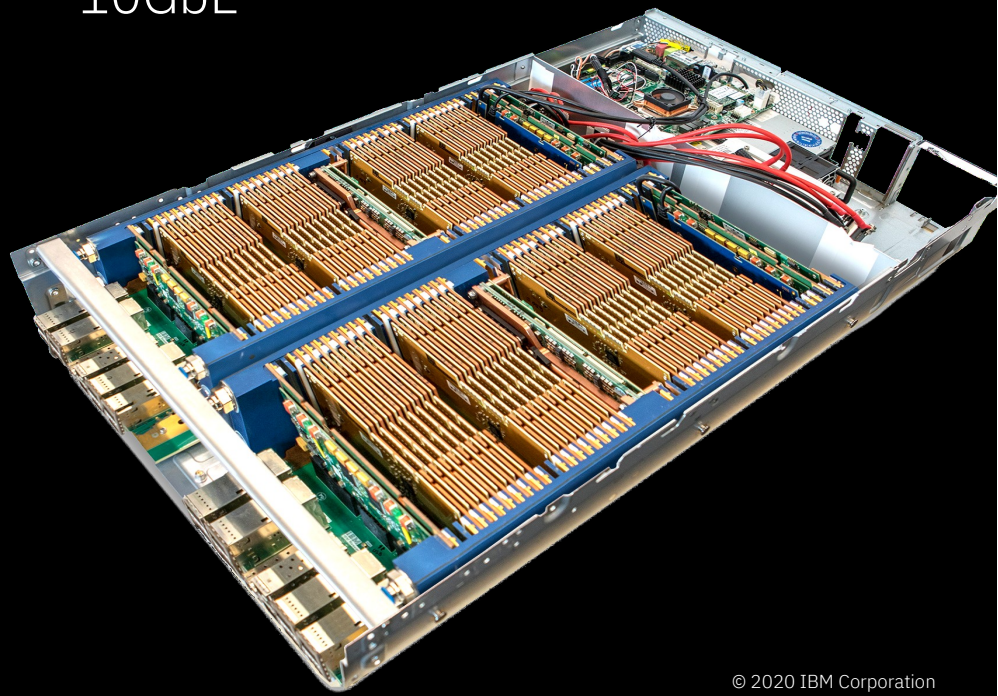
Evaluation: 2D stencil

- CPU: i7 CPU 960 @ 3.20GHz
- simple 2D Jacobi iteration
  - mostly communication bound
- resource usage for 1024x1024 on 8 nodes
  - 1-5% for MPI run-time env. (MPE)

Resource	Available	Used		
		Total	MPE	APP
LUT	331680	113832	1940	1715
LUTRAM	146880	11103	8	0
FF	663360	132553	1744	1058
BRAM	1080	609	3	320

cloudFPGA platform:

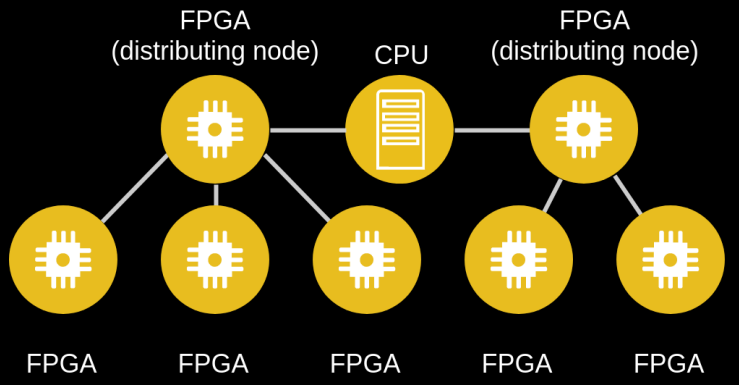
- Kintex KU60 @ 6.4ns
- 64 FPGAs / 2U chassis
- 10GbE



Evaluation: 2D stencil

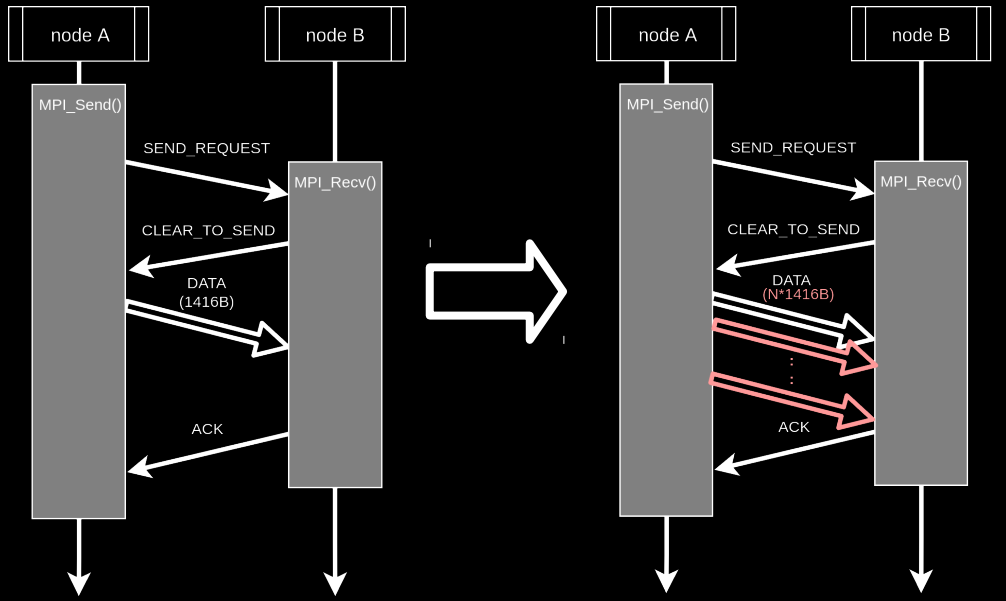
- Power on average (incl. memory & NIC):
  - CPU node 127 W
  - FPGA node 8.5 W
- FPGA speedup: 2.0 – 5.6
- mostly communication bound:
  - halo regions synchronization
  - “unoptimized” protocol
  - for FPGAs earlier than for CPUs
- tree optimization (still communication bound):
  - CPU only: speedup 1.27 due to parallelization
  - ReH²PC: speedup 1.32 due to parallelization and FPGAs are doing most of the communication

Data size	cluster size	CPU only		ReH²PC (1 CPU, size-1 FPGAs)	
		ms / iteration	avg. Watt	ms / iteration	avg. Watt
16x16	2	23.33	254.49	7.59	136.24
256x256	4	1670.90	508.48	444.77	154.21
256x256	8	2136.72	1018.08	842.58	187.68
1024x1024	8	43120.16	1017.99	7763.87	187.09
1024x1024	16	36261.91	2036.16	9311.71	253.23
1024x1024	32	41021.51	4072.33	20939.65	387.99
256x256 (with tree optim.)	8	1686.55	1017.68	637.87	187.97



Future Work

- initial philosophy: *"functionality first, optimized performance later"*
  - common programming model for ReH²PC requires holistic approach
  - learn how to optimize on the way
- example: latest execution time based on improved protocol

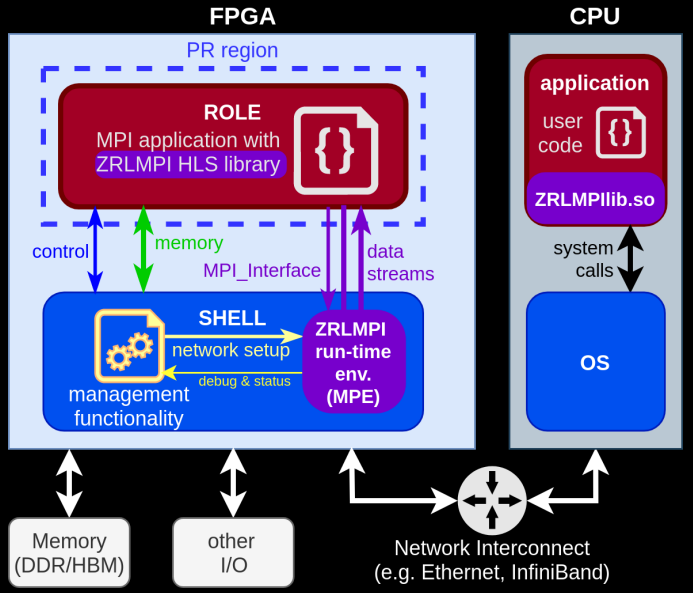


256x256 8 nodes w/ tree-optimization	old ms/iteration	new ms/iteration	speedup
CPU only	1686.55	681.29	2.5
ReH²PC	637.87	34.41	18.6

Summary: MPI for ReH²PC using Transpilation

- ZRLMPI as *proof of concept*
  - exhibit the potential of MPI as standard for ReH²PC
  - full stack solution
  - ...but also full stack to optimize
- Roughly: (early state)

  ZRLMPI on a large ReH²PC cluster:
  - ¼ time (speedup 2.0 – 5.6)
  - 10% of the power (9% – 53%)
- Future Work:
  - optimize tree structure
  - more collectives, memory operations
  - handle non-synthesizable code



Thank you...

ngl@zurich.ibm.com
zurich.ibm.com/cci/cloudFPGA/
@0xcaffee

cloudFPGA: Further Reading

- B. Ringlein, F. Abel, A. Ditter, B. Weiss, C. Hagleitner and D. Fey, “ZRLMPI: A Unified Programming Model for Reconfigurable Heterogeneous Computing Clusters” in 28th IEEE International Symposium On Field-Programmable Custom Computing Machines (FCCM), 2020.
- B. Ringlein, F. Abel, A. Ditter, B. Weiss, C. Hagleitner and D. Fey, “ System architecture for network-attached FPGAs in the cloud using partial reconfiguration,” in 29th International Conference on Field Programmable Logic and Applications (FPL), 2019.
- F. Abel, J. Weerasinghe, C. Hagleitner, B. Weiss, S. Paredes, “An FPGA Platform for Hyperscalers,” in IEEE 25th Annual Symposium on High-Performance Interconnects (HOTI), Santa Clara, CA, pp. 29–32, 2017.
- Weerasinghe, F. Abel, C. Hagleitner, A. Herkersdorf, “Disaggregated FPGAs: Network performance comparison against bare-metal servers, virtual machines and Linux containers,” in IEEE International Conference on Cloud Computing Technology and Science (CloudCom), Luxembourg, 2016.
- J. Weerasinghe, R. Polig, F. Abel, “Network-attached FPGAs for data center applications,” in IEEE International Conference on Field-Programmable Technology (FPT ’16), Xian, China, 2016.
- J. Weerasinghe, F. Abel, C. Hagleitner, A. Herkersdorf, “Enabling FPGAs in hyperscale data centers,” in IEEE International Conference on Cloud and Big Data Computing (CBDCom), Beijing, China, pp. 1078–1086, 2015.
- F. Abel, “How do you squeeze 1000 FPGAs into a DC rack?” online at LinkedIn
- The **cloudFPGA project page at ZRL:** https://www.zurich.ibm.com/cci/cloudFPGA/

References

[1] 53rd TOP500 List - Highlights, June 2019 https://www.top500.org/static/media/uploads/top500_ppt_201906.pdf

Backup

Transpilation of MPI_Scatter

Original

known due to cluster description

known due to constant propagation

```
MPI_Comm_size( MPI_COMM_WORLD, &size );
....
int ldim = DIM/size;
int start = rank * ldim;
....
MPI_Scatter(&grid[0][0], ldim*DIM, MPI_INTEGER, &lgrid[start][0], ldim*DIM, MPI_INTEGER,
                                                                0, MPI_COMM_WORLD);
...
```

original start

datatype

root rank

Transpilation

in case: rank == root rank

```
for(int i = 0; i<SIZE; i++) {
  int chunk_size = ldim*DIM;
  int *new_start = &grid[0][0] + i*chunk_size;
  if(i == 0) {
    memcpy(&lgrid[start][0], new_start,
           chunk_size * sizeof(int));
    continue;
  }
  MPI_Send(new_start, chunk_size, MPI_INTEGER,
           i, SCATTER_TAG, MPI_COMM_WORLD);
}
...
```

otherwise

```
...
MPI_Recv(&lgrid[start][0], ldim*DIM,
         MPI_INTEGER, 0, SCATTER_TAG,
         MPI_COMM_WORLD,
         MPI_STATUS_IGNORE);
...
```